In the Claims:

1. (Currently amended) A computerized method for creating an instrumented executable file, the method comprising:

modifying an executable file to invoke ~~having an original function with~~ a user-supplied function in place of an original function, the user-supplied function injecting a fault into the executable file; and

retaining access information of the original function, the access information enabling the user-supplied function to invoke the original function.

2. (Original) The computerized method for creating an instrumented executable file as in claim 1, wherein the user-supplied function is modified to invoke the original function using the retained access information of the original function.

3. (Previously presented) The computerized method for creating an instrumented executable file as in claim 1, wherein the user-supplied function is in a dynamic link library.

4. (Original) The computerized method for creating an instrumented executable file as in claim 1, wherein the user-supplied function is not exported during compilation.

5. (Original) The computerized method for creating an instrumented executable file as in claim 1, wherein the original function and the user-supplied function have identical prototypes.

6. (Original) The computerized method for creating an instrumented executable file as in claim 1, where the user-supplied function is stored in a module that is separate from the executable file.

7. (Previously presented) The computerized method for creating an instrumented executable file as in claim 1, wherein modifying the executable file is performed using user-specified set points.

8. (Original) The computerized method for creating an instrumented executable file as in claim 7, wherein modifying the executable file further comprises determining whether the original function implements the thiscall calling convention, and when the determination is positive, adding instructions to the executable file to perform:

pushing the register that holds the 'this' pointer onto the stack from the invoked original function site when the determining indicates that the function implements a thiscall calling convention; and

swapping the return value of the invoking original function on the stack and the register that holds the 'this' pointer value on the stack when the determining indicates that the function implements a thiscall calling convention.

9. (Previously presented) The computerized method for creating an instrumented executable file as in claim 7, wherein modifying the executable file further comprises enabling the user-supplied function to invoke the original function in the executable file.

10. (Original) The computerized method for creating an instrumented executable file as in claim 9, wherein enabling the user-supplied function to invoke the original function in the executable file further comprises:

adding a jump in the user-supplied function to a function that retrieves the address of the original function; and

adding a jump in the user-supplied-function that invokes the original function using the address of the original function.

11. (Original) The computerized method for creating an instrumented executable file as in claim 1, further comprising enabling the user-supplied function to alter behavior.

12. (Original) The computerized method for creating an instrumented executable file as in claim 11, wherein enabling the user-supplied function to alter behavior is performed in response to data.

13. (Original) The computerized method for creating an instrumented executable file as in claim 12, wherein the data is retrieved from an initialization file.

14. (Original) The computerized method for creating an instrumented executable file as in claim 1, wherein the retaining further comprises:

saving the address of an original function in a threaded local storage variable; and

creating an entry in a function lookup table associating the address of the original function with the name of the original function, wherein the function lookup table is in the instrumented executable file.

15. (Currently amended) A computerized method for executing an instrumented executable file comprising:

modifying the instrumented executable file to invoke ~~having an original function with~~ a user-supplied function in place of an original function, the user-supplied function injecting a fault into the executable file, the user-supplied function having a jump to the original function;

saving the address of the original function in a threaded local storage variable; and

invoking the user-supplied function using the address.

16. (Original) The computerized method for executing an instrumented executable file as in claim 15, further comprising creating a master lookup table at initialization wherein the master lookup table associates the base address of the instrumented executable file to the address of a function lookup table in the instrumented executable file.

17. (Original) The computerized method for executing an instrumented executable file as in claim 15:

wherein original function is in a dynamic link library; and

wherein the saving and the invoking is performed by a stub function of the original function, the stub function being located in the instrumented executable file.

18. (Original) The computerized method for executing an instrumented executable file as in claim 15:

wherein original function is embedded in the instrumented executable file; and

wherein the saving and the invoking is performed by the original function.

19. (Original) The computerized method for executing an instrumented executable file as in claim 15, further comprising invoking the original function from within the user-supplied function using the threaded local storage variable.

20. (Original) The computerized method for executing an instrumented executable file as in claim 19, wherein invoking the original function further comprises:

pushing the register that holds the 'this' pointer onto the stack from the invoked original function site when the determining indicates that the function implements a thiscall calling convention; and

swapping the return value of the invoking original function on the stack and the register that holds the 'this' pointer value on the stack when the determining indicates that the function implements a thiscall calling convention.

21. (Currently amended) A computerized method for instrumenting an imported function in an executable file for testing by callers of the imported function, the method comprising:

adding a wrapper of the imported function to an import data block;

adding a stub function for the imported function wherein the stub function comprises an instruction that saves the address of the imported ~~import~~ function to a threaded

local storage variable and replaces an access to the imported ~~import~~ function with an access to the user-supplied function, the user-supplied function injecting a fault into the executable file; and

adding an entry in a function lookup table of the imported function.

22. (Currently amended) The computerized method for instrumenting an imported function in an executable file as in claim 21 ~~22~~, the method further comprising:

determining if the prototype of the imported function is correctly specified; and

indicating an error when the determining indicates an incorrectly specified prototype of the imported function.

23. (Currently amended) A computerized method for instrumenting an embedded function in an executable file for testing by callers of the embedded function, the method comprising:

modifying the ~~an~~ embedded function to invoke ~~with~~ a user-supplied function in place of the embedded function using a wrapper, the user-supplied function injecting a fault into the executable file; and

adding an entry in a function lookup table of the address of the embedded function.

24. (Original) A computerized method for instrumenting an embedded function in an executable file as in claim 23, wherein the redirecting is accomplished by an instruction that causes a jump to the user-supplied function.

25. (Original) A computerized method for instrumenting an embedded function in an executable file as in claim 23, the method further comprising:

determining whether the prototype of the embedded function is correctly specified; and

indicating an error when the determining whether the prototype of the embedded function is correctly specified indicates an incorrectly specified prototype of the embedded function.

26. (Original) A computerized method for instrumenting an embedded function in an executable file as in claim 23, wherein the function lookup table is in the executable file.

Claim 27 (Canceled)

28. (Currently amended) A computerized system comprising:

means for modifying an executable file to invoke ~~having an original function with~~ a user-supplied function in place of an original function, the user-supplied function injecting a fault into the executable file; and

means for retaining access information of the original function, the access information enabling the user-supplied function to invoke the original function.

29. (Currently amended) A computerized system comprising:

an executable file having a call to an original function, the original function having an identity comprising a name and a parameter prototype;

means for modifying the executable file to invoke ~~the original function with~~ a user-supplied function in place of an original function, the user-supplied function injecting a fault into the executable file; and

means for configuring the user-supplied function to retrieve stored access information of the original function.

30. (Currently amended) A computerized system comprising:

an executable file having a jump to an original function, the original function having an identity comprising a name and a parameter prototype;

a first software component having a user-supplied function that includes a jump to the original function; and

a second software component for:

receiving the identity of the original function;

receiving the identity of the user-supplied function;

instrumenting the executable file by modifying the executable file to invoke ~~the identity of the original function with~~ the identity of the user-supplied function in place of the identity of the original function, the identity of the user-supplied function injecting a fault into the executable file; and

storing the original function address in the executable file in association with the name of the original instrumented function.

31. (Currently amended) A computerized system comprising:

a first module of machine-readable code comprising:

a call to an original function, the call being directed to a user-supplied function; and

a first data structure associating the identity of the original function with the location of the original function; and

a second module comprising the user-supplied function, linked to the first module and a jump to the original function, the user-supplied function injecting a fault into the machine-readable code.

32. (Original) The computerized system as in claim 31,

wherein the first data structure comprises a function lookup table readily available for verifying that the threaded local storage variable contains the correct original instrumented function address; and

wherein the second module comprises a dynamic linked library.

33. (Original) The computerized system as in claim 31, further comprising a second data structure associating the location of the first data structure with the location of the first module.

Claims 34-35 (Cancelled)

36. (Currently amended) A computer-readable medium having computer-executable instructions to [[a]] cause a computer to perform a method comprising:

modifying an executable file to invoke ~~having an original function with~~ a user-supplied function in place of an original function, the user-supplied function injecting a fault into the executable file; and

retaining access information of the original function, the access information enabling the user-supplied function to invoke the original function.

Claims 37-40 (Cancelled)

41. (Currently amended) A computer-implemented method for configuring an executable file, the executable file having an access to an original function, the computer-implemented method comprising:

replacing the access to the original function with an access to a user-supplied function; and

retaining access information associated with the original function, the access information enabling the user-supplied function to invoke the original function, the user-supplied function injecting a fault into the executable file.

42. (Previously presented) The computer-implemented method of Claim 41, further comprising configuring the user-supplied function to invoke the original function using the access information associated with the original function.

43. (Previously presented) The computer-implemented method of Claim 41, wherein replacing the access to the original function with the access to the user-supplied function is performed by modifying the executable file.

44. (Previously presented) The computer-implemented method of Claim 41, wherein replacing the access to the original function with the access to the user-supplied function

is performed by modifying set points stored in a computer-readable medium separate from the executable file.

45. (Previously presented) The computer-implemented method of Claim 41, wherein retaining access information associated with the original function includes saving the address of the original function.

46. (Previously presented) The computer-implemented method of Claim 41, wherein retaining access information associated with the original function includes associating the name of the original function with the address of original function using a function lookup table.

47. (Previously presented) The computer-implemented method of Claim 46, further comprising invoking the original function using the function lookup table.